ICPC2011 国内予選問題 D　そして、いくつになった？

概要

- 色は 4 色で、同じ色の円盤は高々6 枚しか出現しないので、円盤の枚数は高々24 枚である。
- したがって、ゲームの局面（状態）は、24 ビットで符号化できる（例えば円盤 i を取り除く→ 第 i ビットを 1 にする）
- ゲームでは、2 枚ずつ円盤を取り除くので、初期状態から到達できる状態の数は高々 $2^{24}/2$ ≒ 800 万である。この程度の状態数なら、深さ優先探索で解を探すことが出来ると思われる。（但し、同じ局面（状態）を 2 回以上展開しないようにする必要はある）
- 解の探索途中で、残りの円盤枚数が 1 以下の局面に達したら、それ以上円盤を取り除くことは出来ないので、探索を終了することができる。

［プログラム例］

```
// ICPC 2011 国内予選問題 D
// http://icpc2011.ait.kyushu-u.ac.jp/icpc2011/contest/all_ja.html#section_D
//          Filename =          pd1.c
//          Compile:            cc -O2 pd1.c -lm      // 最適化コンパイルをした方が良い
//          Execution:          ./a.out < D0 > D0.result
//          Check:              diff D0.ans D0.result
// 最大の円盤の枚数は 24 枚なので、24 ビットで円盤の取り除き状況を表現する
// 円盤 i を取り除いた <-> 第 i ビットを 1 にする
// 円盤は2枚ずつ取り除くので状態空間の大きさは約(2^(24))/2 = 800 万状態
// この状態空間を深さ優先探索し、解を求める
// 円盤の残り枚数が 1 以下になった場合は探索を強制終了
// 同じ状態を複数回探索しないように各状態に対する訪問済みフラグを設けて管理する
//
#include <stdio.h>
#include <math.h>

#define MAX_COLORS 4                            // 最大の色番号
#define NPLATE_COLOR 6                          // 同じ色の円盤の最大枚数
#define MAX_N (MAX_COLORS * NPLATE_COLOR)       // 最大の円盤の枚数
#define MAX_STATE (1 << MAX_N)                  // テーブル上の円盤の状態番号の最大値

int n;                          // 円盤の枚数
int x[MAX_N];                   // 円盤の中心の x 座標　0以上100以下
int y[MAX_N];                   // 円盤の中心の y 座標　0以上100以下
int r[MAX_N];                   // 円盤の半径　1以上100以下
int c[MAX_N];                   // 円盤の色番号　1以上4以下
int max_del;                    // 取り除くことができた円盤の枚数の最大値
char visit_flag[MAX_STATE];     // 各状態に対する訪問済みフラグ
int weight[MAX_N];              // 円盤 i のビット位置を整数で表現した配列
```

```
// 状態 sid において 円盤 dn が削除済みなら1を返し、そうでなければ0を返す。
#define IsDeleted(sid, dn)          (((sid) & weight[dn])?1:0)

// 状態 sid に対して、円盤 d1 と円盤 d2 を削除した状態を返す。
#define DelDisks(sid, d1, d2)       ((sid) | weight[d1] | weight[d2])

// 状態 sid を訪問済みなら1を返す。未訪問なら、訪問済みにして0を返す。
#define IsVisited(sid)  ((visit_flag[sid])?1:(visit_flag[sid]=1,0))

// 状態 sid で削除されている円盤の枚数を求める
int num_deleted(int sid)
{
   int c=0;                 // sid を2進数表現したときの1の個数を数える
   while(sid){              // sid = 0 なら数え残しの削除された円盤はない
      if(sid & 1) c++;      // sid の最下位ビットが 1 ならカウントを増やす
      sid = sid >> 1;       // sid を1ビット右へシフト
   }
   return c;                // c が求める値
}

// 状態 sid において、指定した円盤 disk の上に他の円盤がなければ 1, あれば 0 を返す
int is_top(int sid, int disk)
{
   int i, xd, yd;
   if(IsDeleted(sid, disk)) return 0;               // 既に指定の円盤が削除済みの場合は0を返す
   // 他の円盤が乗っている場合、その円盤の番号は disk より小さい
   // disk より番号が小さい円盤を降順にチェックした方が早く判定出来る可能性が高い
   for(i=disk-1; i>=0; i--){
      if(IsDeleted(sid, i)) continue;  // 既に削除された円盤は無視
      xd = x[i]-x[disk];               // 2つの円盤の中心の x 座標の差
      yd = y[i]-y[disk];               // 2つの円盤の中心の y 座標の差
      // 2つの円盤の中心間の距離が、2つの円盤の半径の和よりも小さければ、上に乗っている
      if(r[i]+r[disk] > sqrt(xd*xd+yd*yd)) return 0;
   }
   return 1;                           // 上に乗っている円盤は無いので 1 を返す
}

// 深さ優先探索で、可能な円盤を順次取り除く
void dfs(int sid)                      // sid はテーブル上の円盤の状態を表している
{
   int i, j, n_del;
   if(IsVisited(sid)) return;                   // 状態 sid をチェック済みなら何もしない
   n_del = num_deleted(sid);                    // これまでに削除された円盤の枚数
   if(n_del > max_del) max_del = n_del;         // 必要に応じて最大の削除枚数を更新
   if(n - max_del <= 1) return;                 // テーブル上に残った円盤の枚数が1以下なら探索終了
   for(i=0; i<n; i++){
      if(is_top(sid, i)){                       // 円盤 i の上に他の円盤は乗っていない
         for(j=i+1; j<n; j++){
               if(is_top(sid, j) && c[i] == c[j]){   // 円盤 j の上に他の円盤は乗っておらず
```

```
                                                    // 円盤 i と j の色が等しい時
                dfs(DelDisks(sid, i, j));           // 円盤 i と j を削除して深さ優先探索を継続
                // 深さ優先探索の結果、
                // テーブル上に残った円盤の枚数が1以下の状態に達していれば探索終了
                if(n - max_del <= 1) return;
              }
            }
          }
        }
    }
}

// 配列 visit_flag を未訪問状態に初期化
void init_visit_flag()
{
    int i;
    for(i=0; i < MAX_STATE; i++)
        visit_flag[i] = 0;
}

// 配列 weight の値を設定： weight[i] = (1 << i)
void init()
{
    int i,w;
    w = 1;
    for(i=0; i<MAX_N; i++){
        weight[i] = w;
        w = (w << 1);
    }
}

int main()
{
    int i;
    init();                             // 配列 weight の値を設定： weight[i] = (1 << i)
    while(1){
        scanf("%d", &n);                // n を入力
        if(n==0) break;                 // n == 0 なら終了
        for(i=0; i<n; i++)              // xi, yi, ri, ci を入力
            scanf("%d %d %d %d", &x[i], &y[i], &r[i], &c[i]);
        max_del = 0;                    // 取り除いた円盤の枚数の最大値を0に初期化
        init_visit_flag();              // 全ての状態を未訪問状態にする
        // 1枚も円盤を取り除いていない初期状態から深さ優先探索で円盤を取り除く
        dfs(0);
        printf("%d¥n", max_del);        // 結果を出力
    }
}
```